Lecture
Arrays
Data Structures & Algorithms

Syed Mudassar Alam

Basic Building Blocks

- **Array**
- **Structure**
- **Classes**

Arrays, structures, and classes are collection of data items with the following differences:

Arrays	Structures/Classes
Elements are of the same type.	Elements may be of different type.
Elements are accessed through an index	Elements are accessed through name.

Ordered Lists

- One of the most common data object is an ordered list.
- An ordered list has elements with definite ordering..
- Examples of ordered lists:

1. Months of the years (Jan, Feb, Mar, Apr, May,…., Dec)
2. English Alphabets (A, B, C, …, Z)
3. Words in a sentence ("This is a book on data structures.")
4. Names of the students in a class stored in the order of their roll numbers.

Operations Defined on an Ordered List

1. Find the length of list.
2. Check if the list is empty.
3. Traverse the list in some order.
4. Get the i^{th} element in the list.
5. Change the value of the i^{th} element.
6. Insert a new element at the i^{th} position.
7. Delete the i^{th} element.

Representation of an Ordered List

- The easiest way to represent an ordered list is by an one-dimensional array where we store the i^{th} element of the list in the array element with index i.
- This is called *sequential* or *linear mapping* – mapping a one-dimensional vector onto a one-dimensional space.

Area to be Covered in previous courses

- Single Dimension Array.

- Sorts
- Search

Arrays

- Each element in the array is accessed with reference to its position of location in the array.

- The position is called Index or Subscript. The index of first element is 0 and the index of last element is length-1;

```
Int arr[10];
Char arr[10];
```

Sorting

Insertion Sort

Bubble Sort

Ascending order

10,
11,
12,
13,

Descending order

15,
14,
13,
12,

Advantages

- Arrays can store a large number of values with single name.
- Arrays are used to process many values easily and quickly.
- The value stored in array can be sorted easily.
- A search process can be applied on array easily.

Representation of Single Dimensional Arrays

- Whole array is stored in a single contiguous memory block.

- Address of the first element is called the *base address* and it is also the start address of array.

- The address of the ith element is given by the following formula:

$$Address_i = base\_address + i * size\_of\_element$$

- Arrays are very efficient way of organizing data since accessing array elements requires O(1).

What is an 2-D Array?

- A two-dimensional array consists of both rows and columns of elements. It is essentially a matrix.

- To declare a two-dimensional array, we merely use two sets of square brackets.
  - The first contains the number of rows
  - The second contains the number of columns

```
//Creates a 2D array with 3 rows and 4 columns
int vals[3][4];
```

- Assume that the two dimensional array called **`val`** is declared and looks like the following:

`val`	Col 0	Col 1	Col 2	Col 3
Row 0	8	16	9	52
Row 1	3	15	27	6
Row 2	14	25	2	10

- To access the cell containing 6, we reference **`val[1][3]`**, that is, row 1, column 3.

- Int array [3][3]
- Array declaration can be taken as these are 3 one dimensional array. First array starts with array[0]. Array[0] gives the address of the first row. Array[1] gives the starting address of second row and so on.

Two Dimensional Array

1	5	3	6
3	2	38	64
22	76	82	99
0	106	345	54

User's view (abstraction)

1	5	3	6	3	2	38	64	22	76	82	99	0	106	345	54

System's view (implementation)

Two Dimensional Array

1	5	3	6
3	2	38	64
22	76	82	99
0	106	345	54

User's view (abstraction)

2D array element are store in 2 ways

1.Row major order

2.Column major order

Two Dimensional Array

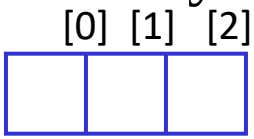
- **Row major – C, Pascal, C++, etc.**
- **Column Major – FORTRAN**

Multi-Dimensional Arrays

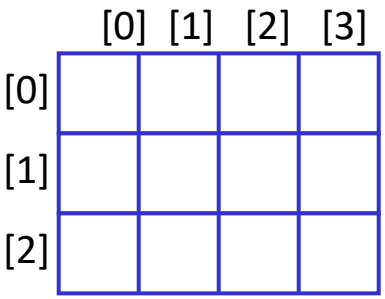
- **Row Major**
  - **Slice along the 1st dimension to get an array of N-1 dimensions**
  - **Continue until you are left with one dimension only.**

do*D1*D2*D3 +d1*D2*D3+d2*D3+d3

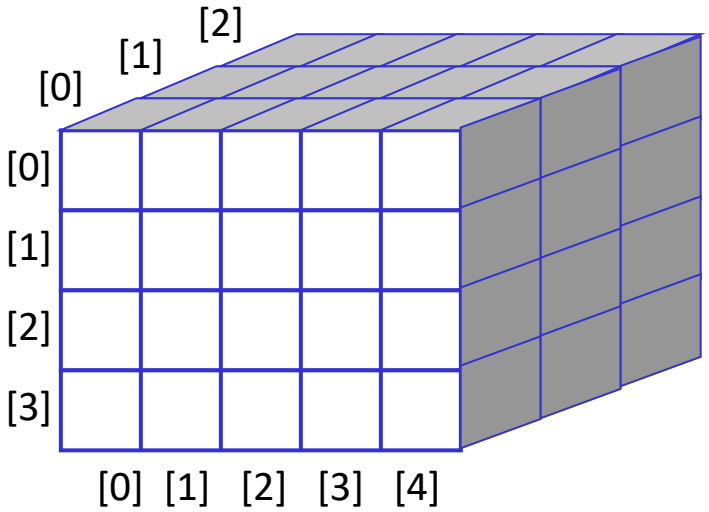
One-dimensional arrays are linear containers.



Multi-dimensional Arrays

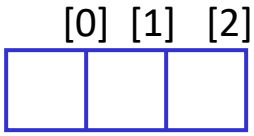


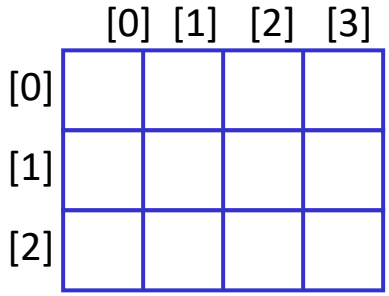
two-dimensional

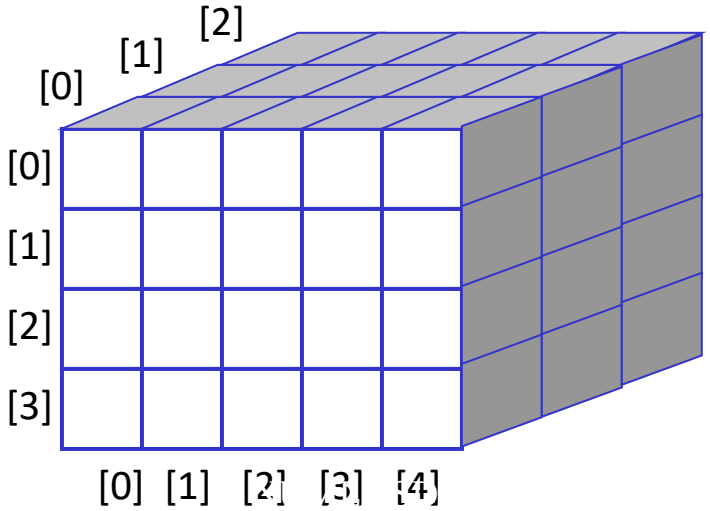


three-dimensional

Initializing Multi-dimensional Arrays







declaration	instantiation
Oval[] **circles**;	**circles = new Oval[3];**
double[][] **table**;	**table = new double[3][4];**
int[][][] **arr**;	**arr = new int[4][5][3];**

- Mapping Function from 2D to 1D
- Array [3][4] //Original
- Array[2][3] //Find?

i * c + j

- 2*4+3=11
- c=no of columns of Original Arrays

Example

```
int main()
{
  const int NUM_ROW(3);
  const int NUM_COL(4);
  int vals[NUM_ROW][NUM_COL] = { {11, 12, 13, 14},
                                 {21, 22, 23, 24},
                                 {31, 32, 33, 34} };

        // output the array
  for (int row = 0; row < NUM_ROW; row++)
    {
      for (int col = 0; col < NUM_COL; col++)
        {
          cout << vals[row][col] << " ";
        }
      cout << endl;
```

Irregular or Jagged Arrays

- Jagged arrays are arrays where each row can have a different number of columns.

- Example:

$$\begin{pmatrix} 2 & 4 & 5 & 0 & & \\ 1 & 12 & & & & \\ 1 & 1 & 10 & 12 & 20 & 30 \end{pmatrix}$$

- How to represent such an array in memory?
- Rows=3 columns=6
- Total space needed?

- 3*6*size of each element

int arr[3][6];

2	4	5	0		
1	12				
1	1	10	12	20	30

Better Representation?

- Use Dynamic Arrays!

- int **arr;
- arr=new int*[3];
- arr[0]=new int[4];
- arr[1]=new int[2];
- arr[2]=new int[6];

How Dynamic Arrays are stored?

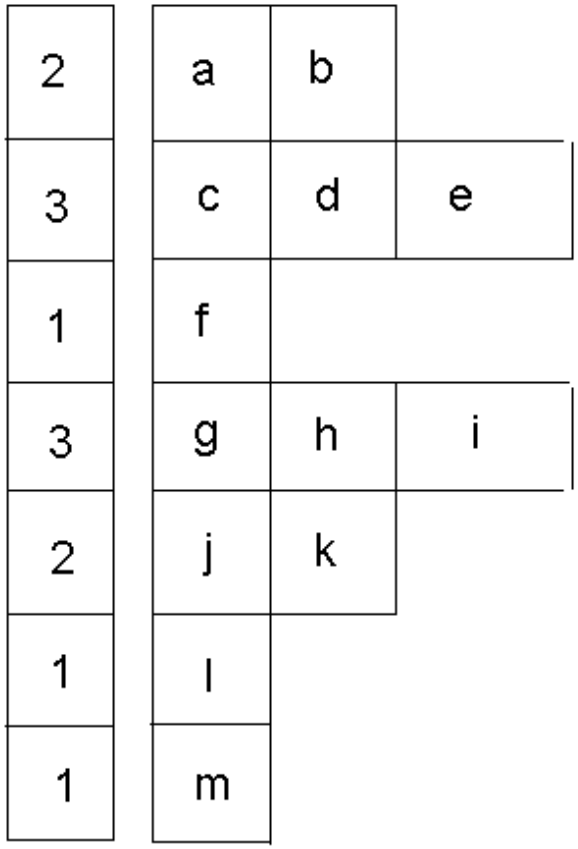
- The bucket locations in individual rows are contiguous, but rows are not necessarily contiguous in heap space.

How about in this case?

```
int arr[][]={{0,2},{5,7,10,12,13},{5},{9,10,12}};
```

- Static arrays not dynamically allocated so how to store them in memory?
- We cannot use simple row major mapping function.
- Number of elements in each row are different!

- If we can find the correct position for the beginning of a particular row, then the mapping function for a one-dimensional array can be used for this portion without any changes.


2
a
b
3
c
d
e
1
f
3
g
h
i
2
j
k
1
l
1
m

Mapping Function

- Find the correct position for the beginning of a particular row, then use the mapping function for a one-dimensional array.

- How?

Arr[i][j]=Base Address + size of each element * $\sum_{k=0}^{i-1} ColArr_k$ + j * size of each element

Triangular Matrix

- What happens in case of a Triangular Matrix?

$$\begin{bmatrix} 1 & 4 & 2 \\ 0 & 3 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

- Row Major order?

NO!

- An upper triangular matrix is mapped using Column Major order.

- Mapping Function?

For all i<=j

a[i][j]=j(j+1)/2 + i

Reading Materiel

Nell Dale: Chapter # 2 (Section 2.1), Chapter # 3
Schaum's Outlines: Chapter # 1
Mark A. Weiss: Chapter # 3 (Section – 3.1, 3.2)